Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Fast cooperative distributed model predictive control based on parametric sensitivity

Tianyu Yu* Zuhua Xu* Jun Zhao* Xi Chen* Lorenz T. Biegler**

** State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, Zhejiang 310027 China (e-mails: yutianyu@zju.edu.cn, zhxu@zju.edu.cn, jzhao@ zju.edu.cn, xi_chen@zju.edu.cn).*
*** Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA, (e-mail: biegler@cmu.edu)}*

Abstract: This paper deals with computational delay in distributed nonlinear model predictive control. A fast, cooperative distributed model predictive control algorithm is proposed based on parametric sensitivity. The implementation strategy is divided into two different stages. In the background stage, the future state is estimated one step forward with the current state and input. The local MPC controllers perform distributed optimization based on the predicted state and iterate to obtain the nominal optimal solutions. In the online stage, all the controllers correct their nominal optimal inputs through parametric sensitivity. Specifically, each controller formulates its local sensitivity equation based on the state estimation error. In order to solve these linear equations in a distributed way, Jacobi iterative method is applied. The overall algorithm can provide fast control action. A case study is provided to show the promising performance of the proposed method.

Keywords: Distributed control, cooperative control, predictive control, nonlinear control, parametric sensitivity.

1. INTRODUCTION

Industrial processes usually consist of several interconnected units that interact with each other through mass, energy and information flows. For these processes, centralized control is impractical due to expensive computational cost, poor reliability and maintainability. Distributed control is a preferable choice in which several controllers are used and communication is allowed among controllers to improve decision making. Flexibility, scalability and robustness are potential advantages for distributed control (Mayne, 2014).

Model predictive control (MPC) is a common control strategy in industry because it can handle operational constraints and integrate various kinds of models. Due to strong nonlinearity of industrial systems, it is necessary to use nonlinear model predictive control (NMPC), which accommodates nonlinear model in MPC optimization for better performance concerns. In this work, distributed implementation of nonlinear model predictive control is studied.

A distributed nonlinear model predictive control (DNMPC) strategy is said to be cooperative if each controller optimizes with a system-wide objective function and cooperates with others to reach a common goal (Christofides et al., 2013). In contrast, non-cooperative approach refers to the control strategy when each controller uses a local control objective (Scattolini, 2009). Over the past years, several DNMPC strategies have been proposed. Dunbar (2007) proposed a non-cooperative DNMPC algorithm for continuous-time systems with analysis of feasibility and stability properties. A distributed nonlinear optimal control strategy is presented in Necoara et al. (2009) with application of sequential convex programming and smoothing techniques, and iteration is needed for optimality. A two-tier DNMPC algorithm is proposed based on Lyapunov MPC techniques (Liu et al., 2009). In this approach, one controller is designed for closed-loop system stabilization and the other is used for optimizing manipulated variables. As an extension of the work, sequential and iterative DNMPC architectures are proposed in Liu et al. (2010). The former utilizes a single-direction and non-iterative computation strategy, while the latter applies a bi-direction and iterative strategy for distributed optimization. A cooperative DNMPC strategy is proposed in Stewart et al. (2011), where a novel distributed optimization procedure is developed for solving non-convex nonlinear programming (NLP) problems. Asymptotic stability of the closed-loop system is ensured.

The use of nonlinear model in NMPC could cause the optimization problem to be non-convex and therefore time-consuming to solve. Although distributed control structure reduces the optimization problem size for each NMPC controller, the inherent difficulty of solving non-convex NLPs still exists, which leads to a delay of input implementation. According to Findeisen and Allgöwer (2004), the computational delay will decrease the control performance and may even destabilize the closed-loop system. To handle this problem, several fast NMPC algorithms have been proposed (Wolf and Marquardt, 2016). Diehl et al. (2005) proposed a real-time iteration (RTI) algorithm for NMPC, which prepares an initial value for input through successive linearization offline, and solves one quadratic programming problem online to obtain fast feedback. A low latency output feedback model predictive control is proposed by Kogel and Findeisen (2017)

for constrained linear systems, where an optimal control problem is solved in background and a correction scheme is triggered online with explicit MPC technique. In particular, a parametric sensitivity-based NMPC approach is proposed in Zavala and Biegler (2009). The algorithm predicts future state in advance and solves the next-step optimization problem in background, and updates the background solution online through sensitivity. This approach can provide fast control actions due to the linear feature of online update strategy. The prediction-correction method in input calculation can be carried over in distributed implementation of NMPC to speed up the online computation. In our previous work, a hierarchical distributed NMPC algorithm is proposed based on sensitivity theory (Yu et al., 2019). The lower-layer local controllers compute the predicted optimal inputs in background via distributed optimization, and the upper-layer coordinator collects and combines all the local optimality information to form the system-wide sensitivity equation. After solution of this equation, the input correction vectors are obtained and the predicted optimal inputs are updated before implementation.

In this paper, a cooperative distributed NMPC algorithm is further proposed based on parametric sensitivity. A two-stage approach is applied to reduce the online computation delay. Different from our previous work (Yu et al., 2019), the proposed method does not require an upper-layer coordinator and is fully distributed. In the background stage, future state is predicted one step forward based on current state and input. Then the local controllers solve their NMPC problems in a distributed manner, and iterate with each other to improve decision making. Nominal optimal local input is obtained for each controller. In the online stage, true state is measured. All the MPCs solve their local sensitivity equations in parallel with the application of Jacobi iterative method. After that, the correction vectors can be computed and the nominal inputs are updated before implementation.

The paper is organized as follows. Section 2 presents an overview of the proposed algorithm first. Then the algorithmic details including the background and online stages of the sensitivity-based DNMPC strategy is given. Section 3 demonstrates a case study on the quadruple-tank plant. Finally, Section 4 concludes the paper.

2. ALGORITHM DESCRIPTION

For a conventional MPC controller, the input is obtained online only after solving the optimization problem. As the NLP problem may occupy plenty of computational time, this control strategy could lead to a substantial time delay of input implementation. As a result, the controller performance may degrade. To speed up control feedback, we propose a two-stage DNMPC algorithm, whose timeline is shown in Fig. 1.

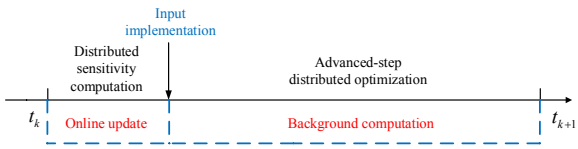

Fig. 1. Timeline for the proposed algorithm.

At sampling time t_k, the online stage is activated. At that time, a pre-calculated input is already available. The goal of this stage is to modify the candidate input before being implemented. First, the true state is measured and the state prediction error can be computed. After that, each MPC controller formulates its local sensitivity equations. Then all the controllers solve these equations in a distributed way to calculate the input correction vectors. Specifically, Jacobi iterative method is applied during the computation. Finally, the nominal optimal inputs are modified with the correction terms, and the corrected inputs are injected to the plant.

After input implementation of the current sampling time, the controller enters the background stage. In this stage, the controller prepares a candidate input for the next sampling time. First, the future state is calculated in advance based on the current state and input. After that, all the local controllers perform distributed optimization iteratively. At each iteration, each controller communicates with others to improve the control action. After distributed optimization converges, the nominal optimal inputs are obtained, and all the optimality information from local controllers is stored in background. The information is prepared for the online computation of next sampling time t_{k+1}.

The proposed method is called advanced-step distributed NMPC (as-DNMPC). In this method, the sensitivity update step that solves linear systems is the only online computation task. Compared with the conventional approach, the proposed method has much less online computational cost.

2.1 Background computation

In as-DNMPC, there are several computational tasks in the background stage. After input implementation of the current sampling time t_k, the future state $x(k+1|k)$ is estimated immediately based on the latest state $x(k)$ and input $u(k)$:

$$x(k+1|k) = f\big(x(k), u(k)\big) \tag{1}$$

where $f(\cdot,\cdot)$ is the nominal system-wide plant model. For cooperative distributed NMPC, each controller has full knowledge of the overall system dynamics. Each MPC performs optimization with an overall objective function and make predictions based on the system-wide plant model. The optimized variable is the local input of the corresponding subsystem. Suppose the subsystem dynamics are coupled through inputs, the prediction model for controller $i = 1, \dots, M$ is:

$$x(k+1) = f\big(x(k), u_i(k), u_{-i}(k)\big) \tag{2}$$

where $u_{-i} = [u_1^T, \dots, u_{i-1}^T, u_{i+1}^T, \dots, u_M^T]^T$ is the coupled input sequence from other subsystems. For controller i, the only optimized input is u_i, while the coupling term u_{-i} is fixed as a constant parameter during the optimization.

Based on the prediction of (1), all the controllers solve their NMPC problems individually, and iteration is needed for distributed optimization. At iteration p, the optimization problem for controller i is:

$$\min_{u_i(k+j|k)} J = \sum_{j=0}^{N-1} l(x(k+j\,|\,k), u(k+j\,|\,k)) + V_f(x(k+N\,|\,k))$$
$$s.t. \begin{cases} x(k+1\,|\,k) = x_0 \\ x(k+j+1\,|\,k) = f(x(k+j\,|\,k), u_i(k+j\,|\,k), u_{-i}^{p-1}(k+j\,|\,k)), j = 1,\dots,N \\ u_i(k+j\,|\,k) \in U_i, j = 1,\dots,N \end{cases} \quad (3)$$

where $l(\cdot,\cdot)$ and $V_f(\cdot)$ denote the stage cost and terminal cost, respectively. After solving (3), controller i can obtain the optimal local input of current iteration, $u_i^{p,*}$. Then all the controllers exchange their latest optimal solutions. The final local input is formulated by convex combination:

$$u_i^p = u_i^{p-1} + w_i(u_i^{p,*} - u_i^{p-1}) \quad (4)$$

in which u_i^p is the final input of controller i and w_i is the weight for controller i, satisfying $w_i \geq 0$ and $\sum_{i=1}^{M} w_i = 1$. The controller weights are chosen properly by a recursive strategy as stated in Section 2 of (Stewart et al., 2011).

Based on the local inputs, the overall input u^p is formulated as $u^p = \left[u_1^{p,T}, \dots, u_M^{p,T}\right]^T$. After each iteration, we check the following stopping criterion:

$$\|u^p - u^{p-1}\| < \varepsilon \quad (5)$$

where ε is a fixed value. If (5) is satisfied, then convergence is reached. Otherwise, the iterative distributed optimization continues. For iteration $p+1$, the coupling input u_{-i}^p is set as the coupling term of the previous iteration, u_{-i}^{p-1}.

According to Section 2 of (Stewart et al., 2011), the distributed optimization algorithm is able to converge after a sufficient number of iterations. However, it may be time-demanding to finish the computation. For practical concerns, we set a maximum iteration number p_{max} to limit the background computational time. If the optimization procedure converges within p_{max} iterations, the final local input u_i^p is set as the nominal optimal solution $u_{i,nom}$ for controller i. Otherwise, the latest local input $u_i^{p_{max}}$ is selected as the optimum.

After the optimization procedure completes, all the controllers preserve their optimality information to prepare for the online stage. The information includes primal and dual variables and the corresponding multipliers in each local controller's Karush-Kuhn-Tucker (KKT) conditions. They are used to compute the Hessian and constraint Jacobian matrices, which formulate the sensitivity equations of NMPC controllers, as discussed below.

2.2 Online update

In this stage, all the computation tasks are performed online. At a new sampling time t_{k+1}, the true system state $x(k+1)$ can be measured, and the state prediction error Δx_0 can be computed by the following equation:

$$\Delta x_0 = x(k+1) - x(k+1|k) \quad (6)$$

Then all the local controllers iteratively solve their sensitivity equations to compute the correction step for the local inputs. For ease of understanding, the basic theory of parametric sensitivity is briefly introduced as follows. Then the analysis of the sensitivity for local controllers will be given.

The NMPC problem (3) can be represented as the following parametric programming problem:

$$\min_s f(s,q)$$
$$s.t. \begin{cases} c(s,q) = 0 \\ s \geq 0 \end{cases} \quad (7)$$

where s is the optimized variable (primal variable) and q is the fixed parameter. According to sensitivity theory, the optimal solution s^* is an implicit function of the parameter q and can be represented as $s^*(q)$ (Fiacco, 1983). For problem (7), the optimal solution $s^*(q_0)$ can be computed with the nominal parameter q_0. When the parameter changes to q_1, an approximate solution of the true optimum $s^*(q_1)$ can be obtained by taking a correction step based on the nominal solution through parametric sensitivity. The sensitivity equation for (7) is:

$$\begin{bmatrix} H & A & -I \\ A^T & 0 & 0 \\ V & 0 & S \end{bmatrix} \begin{bmatrix} \Delta s \\ \Delta\lambda \\ \Delta v \end{bmatrix} = - \begin{bmatrix} \nabla_{s,q} L \\ \nabla_q c^T \\ 0 \end{bmatrix} \cdot \Delta q \quad (8)$$

where λ is the equality multiplier, v is the bound multiplier, L is the Lagrangian function, H is the Hessian matrix, A is the constraint Jacobian matrix, $V = diag(v)$ is the bound multiplier matrix and $S = diag(s)$ is the primal variable matrix. $\Delta q = q_1 - q_0$ is the perturbation step of parameter. These matrices and the sensitivity vectors $\nabla_{s,q} L$, $\nabla_q c^T$ can be computed based on the nominal solution $s^*(q_0)$.

After solving equation (8), we can extract Δs from the solution vector $\Delta d = [\Delta s^T, \Delta\lambda^T, \Delta v^T]^T$ and make the following correction:

$$\tilde{s}(q_1) = s^*(q_0) + \Delta s \quad (9)$$

where $\tilde{s}(q_1)$ is the first-order approximation of the true optimum $s^*(q_1)$. Since (8) is a linear system, the computational cost for sensitivity approximation is very cheap.

In our case, problem (3) is parametric in the initial state x_0 and the coupled input sequence $u_{-i}^{p-1}(k+j|k)$. According to sensitivity theory, the perturbation steps of these two parameters, Δx_0 and Δu_{-i}, should appear in the sensitivity equation. The sensitivity equation for local controller i is:

$$\begin{bmatrix} H_i & A_i & -I \\ A_i^T & 0 & 0 \\ V_i & 0 & S_i \end{bmatrix} \begin{bmatrix} \Delta s_i \\ \Delta\lambda_i \\ \Delta v_i \end{bmatrix} = - \begin{bmatrix} \nabla_{s_i,x_0} L \\ \nabla_{x_0} c^T \\ 0 \end{bmatrix} \cdot \Delta x_0 - \begin{bmatrix} \nabla_{s_i,u_{-i}} L \\ \nabla_{u_{-i}} c^T \\ 0 \end{bmatrix} \cdot \Delta u_{-i} \quad (10)$$

Where $\Delta u_{-i} = [\Delta u_1^T, \dots, \Delta u_{i-1}^T, \Delta u_{i+1}^T, \dots, \Delta u_M^T]^T$ and $\Delta s_i = [\Delta x^T, \Delta u_i^T]^T$. Here, we denote $J_{i,-i} = \nabla_{s_i,u_{-i}} L$, $N_{i,-i} = \nabla_{u_{-i}} c^T$ and $G_i = \nabla_{x_0} c^T$. In fact, $J_{i,-i}$ and $N_{i,-i}$ are the coupling matrices among subsystems, which can be computed in the background stage based on the dynamic equations. Note that $\nabla_{s_i,x_0} L = 0$, (10) can be represented as:

$$\begin{bmatrix} H_i & A_i & -I \\ A_i^T & 0 & 0 \\ V_i & 0 & S_i \end{bmatrix} \begin{bmatrix} \Delta s_i \\ \Delta\lambda_i \\ \Delta v_i \end{bmatrix} = - \begin{bmatrix} 0 \\ G_i \\ 0 \end{bmatrix} \cdot \Delta x_0 - \begin{bmatrix} J_{i,-i} \\ N_{i,-i} \\ 0 \end{bmatrix} \cdot \Delta u_{-i} \quad (11)$$

In this equation, all the matrices can be computed in the background stage after solving problem (3). Once the true state $x(k+1)$ is measured, the initial state perturbation Δx_0 can be computed. However, for controller i, the coupled input perturbation Δu_{-i} is unknown. Therefore, we need to assume a certain value for Δu_{-i} in solving (11).

To solve all the local sensitivity equations, Jacobi iterative method is applied. At iteration r, controller i solves the following equation assuming that Δu_{-i} remains at the value of its previous iteration:

$$\begin{bmatrix} H_i & A_i & -I \\ A_i^T & 0 & 0 \\ V_i & 0 & S_i \end{bmatrix} \begin{bmatrix} \Delta s_i^r \\ \Delta \lambda_i^r \\ \Delta v_i^r \end{bmatrix} = - \begin{bmatrix} 0 \\ G_i \\ 0 \end{bmatrix} \cdot \Delta x_0 - \begin{bmatrix} J_{i,-i} \\ N_{i,-i} \\ 0 \end{bmatrix} \cdot \Delta u_{-i}^{r-1} \quad (12)$$

At each iteration, all the controllers solve their local sensitivity equations in parallel. After solution, we formulate the overall solution vector $\Delta d^r = \left[\Delta d_1^{r,T}, \dots, \Delta d_M^{r,T}\right]^T$ from (12) and check the following condition:

$$\|\Delta d^r - \Delta d^{r-1}\| < \delta \quad (13)$$

where $\delta > 0$ is a constant and $\Delta d_i^r = \left[\Delta s_i^{r,T}, \Delta \lambda_i^{r,T}, \Delta v_i^{r,T}\right]^T$. If (13) is satisfied, the Jacobi algorithm converges. Otherwise, we update the value of Δu_{i-1} and re-solve (12). For real-time implementation concerns, we set a maximum iteration number r_{max} for the Jacobi method. If the method does not converge within r_{max} iterations, the computation procedure will be stopped, and the latest solution vector $\Delta d^{r_{max}}$ is taken as the final solution.

Remark 1. The convergence of the Jacobi iterative method depends on the values of the matrices H_i, A_i, V_i, S_i , $J_{i,-i}$, $N_{i,-i}$. All the existing theorems in this field can be used to check the convergence of the algorithm. Due to space limitations, we will not go into the details here.

When the computation procedure terminates, all the controllers take the latest solution Δu_i^r from Δd^r as the input correction vector Δu_i. The nominal optimal local input is corrected thereby:

$$u_{i,new} = u_{i,nom} + \Delta u_i \quad (14)$$

In this stage, local sensitivity equations are formulated for all controllers by applying parametric sensitivity to their NMPC problems. Jacobi iterative method is performed solve these linear systems, and the input correction vectors can be obtained from the solution vectors. The nominal local inputs are updated online based on the correction and then implemented to the plant.

2.3 Algorithm procedure

The proposed algorithm applies a two-stage strategy. During the background stage, there are three computational tasks, listed as follows.

(a) Advanced-step prediction: Set $p = 0$. Estimate the future state $x(k+1|k)$ with the current state $x(k)$ and input $u(k)$. Set the estimated state as the initial state for the optimization problem of all local controllers.

(b) Iterative distributed optimization: At iteration p, each controller solves its NMPC problem (3). After that, controller i obtains the optimal local input $u_i^{p,*}$. Then formulate the final local input u_i^p by convex combination. Check stopping criterion (5) each time until it is satisfied or the maximum iteration number p_{max} is reached. Store the latest solution as the nominal optimal local input $u_{i,nom}$.

(c) Optimality information collection: Collect primal and dual variables of problem (3) for each controller. Compute all the matrices and sensitivity vectors in (11) based on the nominal solution and store them in the background.

During the online stage, there are three steps as follows.

(a) Measure or estimate the actual system state $x(k+1)$. Compute the state prediction error.

(b) Solve local sensitivity equations for all controllers, based on the assumption that the coupled terms are fixed, and apply Jacobi iterative method to reach convergence. If the algorithm converges or reaches the maximum iteration number, the computation procedure stops. After that, extract the local input correction step Δu_i for each system from the solution vector Δd_i.

(c) Each controller updates its nominal local input by (14), and implements the updated input $u_{i,new}$ to the system. Set $k = k+1$ and return to the background stage.

3. CASE STUDY

We apply the proposed algorithm to the quadruple-tank plant, with its diagram shown in Fig. 2. In this plant, the bottom liquid is pumped to the four tanks by two pumps. For each stream, a fixed portion of the flow is directed into one lower tank and the rest is directed into the upper tank on the opposite side. In addition, the two upper tanks discharge into the two corresponding lower tanks.

The plant is divided into two subsystems based on relative gain array analysis in (Alvarado et al., 2011). The states of subsystem 1 are the liquid levels of Tank 1 and Tank 3, and the input is the flow rate of Pump 2. The states of subsystem 2 are the liquid levels of Tank 2 and Tank 4, and the input is the flow rate of Pump 1.

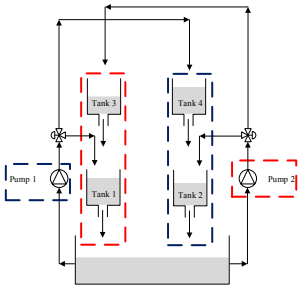

Fig. 2. Diagram of the quadruple-tank plant.

The dynamics of the plant can be described by the following equations:

$$\begin{cases} \dfrac{dh_1}{dt} = -K_1\sqrt{h_1} + K_2\sqrt{h_3} + K_3u_1 \\ \dfrac{dh_2}{dt} = -K_4\sqrt{h_2} + K_5\sqrt{h_4} + K_6u_2 \\ \dfrac{dh_3}{dt} = -K_2\sqrt{h_3} + K_7u_2 \\ \dfrac{dh_4}{dt} = -K_5\sqrt{h_4} + K_8u_1 \end{cases} \quad (15)$$

where $h_i,\ i = 1,\dots,4$ represents the liquid levels of the four tanks and $u_i,\ i = 1,2$ are the flow rates of the two pumps. The values of the parameters $K_i,\ i = 1,\dots,8$ can be found in (Yu et al., 2019).

The control problem is to track piecewise constant setpoints for the liquid levels of the lower two tanks. Therefore, the liquid levels of Tank 1 and Tank 2 are the outputs of the plant. In this study, the tracking problem is transformed into a regulation problem based on different steady-state values of the setpoints. Then a step response test is performed to observe the system dynamics. The sampling time of the plant is set as 20s. For each controller, the same system-wide objective function is used. For the stage cost matrices, $Q = diag(10,10,0,0)$ and $R = diag(0.01,0.01)$. The terminal cost weight matrix $P_f = diag(10,10,0,0)$. The prediction horizon $P = 50$ and the control horizon $M_c = 5$. Each local controller satisfies $u_i(k+j|k) = u_i(k+M_c-1|k)$ for $i = 1,\dots,M$ and $j = M_c,\dots,P-1$. For background computation, the maximum iteration number p_{max} is 100 and the constant is $\varepsilon = 10^{-2}$. For Jacobi method, the maximum iteration number r_{max} is 10 and the constant $\delta = 10^{-3}$.

To demonstrate the control performance, the plant-model mismatch is introduced by perturbing the parameter K_3 from its nominal value $K_{3,nom}$. We set the perturbed values of K_3 as $0.5K_{3,nom}$, $0.65K_{3,nom}$, $0.8K_{3,nom}$, respectively. Two algorithms, ideal DNMPC and as-HDNMPC, are used for comparison. Ideal DNMPC uses a standard implementation approach, which computes the optimal control law online based on the current measurement. The control input is injected to the plant until optimization is finished. The control algorithm proposed in Yu et al. (2019) is named as-HDNMPC. The output profiles of Tank 1 and Tank 2 under different magnitudes of mismatch are shown in Figs. 3 and 4, respectively. The time indices of the proposed algorithm under different cases are shown in Table 1.

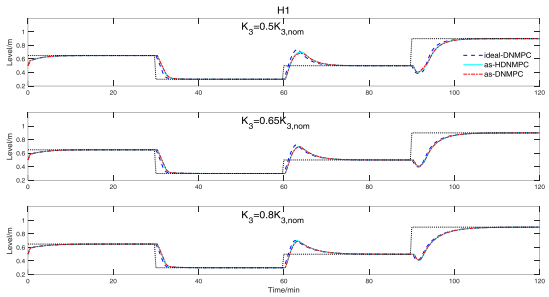

Fig. 3. Liquid level of Tank 1 under different mismatch cases.

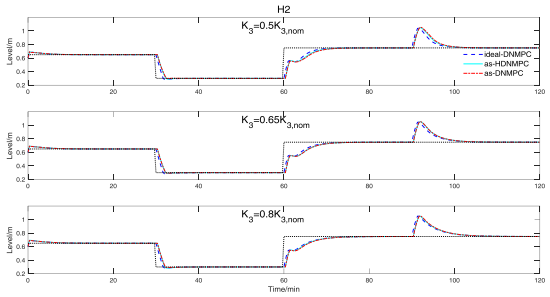

Fig. 4. Liquid level of Tank 2 under different mismatch cases.

Table 1. Time indices of as-DNMPC under different magnitudes of mismatch.

Value of $K_3/K_{3,nom}$	Average online computational time/CPUs	Maximum online computational time/CPUs	Maximum iteration number
0.5	0.0582	0.1248	3
0.65	0.0548	0.1248	3
0.8	0.0761	0.1560	2

As shown in Figs. 2 and 3, all algorithms are able to achieve setpoint tracking in all cases. The two sensitivity-based algorithms produce almost identical control action, as they have similar procedure for input computation. The output profiles of the proposed algorithm is a little lagging compared to that of ideal DNMPC, due to the suboptimal solution obtained by the sensitivity-based strategy. Table 3 shows that the average computational time is less than 0.1 CPUs in all cases. Moreover, the maximum time is less than 0.2 CPUs under different magnitudes of mismatch. As for the iteration number, it requires 3 iterations to solve sensitivity equations at most, which is much smaller than the maximum iteration number r_{max}. This implies that the Jacobi method is able to converge in a few iterations. Therefore, the proposed strategy can provide time-critical control actions.

We then focus on the mismatch scenario when the perturbed parameter $K_3 = 0.8K_{3,nom}$, and compare the time indices with ideal DNMPC algorithm. Table 2 shows the time indices of the two algorithms.

Table 2. Time indices for each algorithm.

Algorithm	Average online computational time/CPUs	Maximum online computational time/CPUs
As-DNMPC	0.0542	0.0936
As-HDNMPC	0.0681	0.1716
Ideal-DNMPC	0.3143	4.5396

Table 2 shows that the average online computational time of as-DNMPC is about 24.2% of the time spent in ideal-DNMPC. As for the maximum online computational time, ideal-DNMPC reaches 4.5396 CPUs, which is about 29 times of that of ideal-DNMPC. For the ideal control strategy, the online computational cost is large as distributed optimization is performed online. In contrast, the proposed algorithm only needs to solve a set of sensitivity equations online. As seen in Table 1, only 2 Jacobi iterations are required to solve sensitivity equations. Compares with our previous work, the newly developed algorithm requires less computational time for input computation. This is because all the controllers solve

their local sensitivity equations in parallel, and convergence can be achieved with few iterations. Therefore, the proposed strategy can provide time-critical control actions.

Remark 2. In all mismatch cases, the background distributed optimization is able to converge within p_{max} iterations. In other words, the maximum iteration number is not activated at any sampling time.

4. CONCLUSIONS

In this paper, a fast two-stage cooperative DNMPC algorithm is proposed to reduce the computational delay in MPC implementation. In the background stage, the future state is estimated in advance based on the nominal model. Each local controller then solves its local NMPC problem and communicates with others to improve the decision making. The nominal optimal input sequence is obtained at convergence. In the online stage, the state prediction error is calculated after new measurement. Then each controller solves its sensitivity equation with Jacobi iterative method to compute the input correction vector. The nominal optimal input is updated online and implemented to the plant. A case study demonstrates good performance of the method.

5. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support of NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization (No. U1509209) and National Key Research and Development Program (No. 2017YFE0106700).

REFERENCES

Alvarado, I., Limon, D., Muñoz de la Peña, D., Maestre, J.M., Ridao, M.A., Scheu, H., Marquardt, W., Negenborn, R.R., De Schutter, B., Valencia, F., Espinosa, J. (2011). A comparative analysis of distributed MPC techniques applied to the HD-MPC four-tank benchmark. *Journal of Process Control*, 21, 800-815.

Christofides, P.D., Scattolini, R., Muñoz de la Peña, D., Liu, J. (2013). Distributed model predictive control: A tutorial review and future research directions. *Computers and Chemical Engineering*, 51, 21-41.

Dunbar, W.B. (2007). Distributed receding horizon control of dynamically coupled nonlinear systems. *IEEE Transactions on Automatic Control*, 52 (7), 1249-1263.

Diehl M., Bock H.G., Schlöder J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43, 1714-1736.

Fiacco, A.V. (1983). Introduction to Sensitivity and Stability Analysis in Nonlinear Programming. Academic Press, New York.

Findeisen, R., Allgöwer, F. (2004). Computational delay in nonlinear model predictive control. *IFAC Proceedings Volumes*, 37 (1), 427-432.

Kogel M., Findeisen R. (2017). Low latency output feedback model predictive control for constrained linear systems. In *Proceedings of 2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 1925-1932. Melbourne, Australia.

Liu, J., Muñoz de la Peña, D., Christofides, P.D. (2009). Distributed model predictive control of nonlinear process systems. *AIChE Journal*, 55 (5), 1171-1184.

Liu, J., Chen, X., Muñoz de la Peña, D., Christofides, P.D. (2010). Sequential and Iterative Architectures for Distributed Model Predictive Control of Nonlinear Process Systems. *AIChE Journal*, 56 (8), 2137-2149.

Mayne, D.Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 50, 2967-2986.

Necoara, I., Savorgnan, C., Dinh, Q.T., Suykens, J., Diehl, M. (2009). Distributed nonlinear optimal control using sequential convex programming and smoothing techniques. In *Proceedings of Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, 543-548. Shanghai, China.

Scattolini, R. (2009). Architectures for distributed and hierarchical Model Predictive Control – A review. *Journal of Process Control*, 19, 723-731.

Stewart, B.T., Wright, S.J., Rawlings, J.B. (2011). Cooperative distributed model predictive control for nonlinear systems. *Journal of Process Control*, 21, 698-704.

Wolf, I.J., Marquardt, W. (2016). Fast NMPC schemes for regulatory and economic NMPC – A review. *Journal of Process Control*, 44, 162–183.

Yu, T., Zhao, J., Xu, Z., Chen, X., Biegler, L.T. (2019) Sensitivity-based hierarchical distributed model predictive control of nonlinear processes. *Journal of Process Control*, 84, 146-167.

Zavala, V.M., Biegler, L.T. (2009). The advanced-step NMPC controller: Optimality, stability and robustness. *Automatica*, 45, 86-93.